An Algebraic Notion of Conditional Independence, and Its Application to Knowledge Representation

Jesse Heyninck
Open Universiteit, the Netherlands
University of Cape Town, South-Africa
jesse.heyninck@ou.nl

Abstract

Conditional independence is a crucial concept supporting adequate modelling and efficient reasoning in probabilistics. In knowledge representation, the idea of conditional independence has also been introduced for specific formalisms, such as propositional logic and belief revision. In this paper, the notion of conditional independence is studied in the algebraic framework of approximation fixpoint theory. This gives a language-independent account of conditional independence that can be straightforwardly applied to any logic with fixpoint semantics. It is shown how this notion allows to reduce global reasoning to parallel instances of local reasoning, leading to fixed-parameter tractability results. Furthermore, relations to existing notions of conditional independence are discussed and the framework is applied to normal logic programming.

1 Introduction

Over the last decades, conditional independence was shown to be a crucial concept supporting adequate modelling and efficient reasoning in probabilistics (Pearl, Geiger, and Verma 1989). It is the fundamental concept underlying network-based reasoning in probabilistics, which has been arguably one of the most important factors in the rise of contemporary artificial intelligence. Even though many reasoning tasks on the basis of probabilistic information have a high worst-case complexity due to their semantic nature, network-based models allow an efficient computation of many concrete instances of these reasoning tasks thanks to local reasoning techniques. Therefore, conditional independence has also been investigated for several approaches in knowledge representation, such as propositional logic (Darwiche 1997; Lang, Liberatore, and Marquis 2002), belief revision (Kern-Isberner, Heyninck, and Beierle 2022; Lynn, Delgrande, and Peppas 2022) and conditional logics (Heyninck et al. 2023). For many other central formalisms in KR, such a study has not yet been undertaken.

Due to the wide variety of formalisms studied in knowledge representation, it is often beneficial yet challenging to study a concept in a language-independent manner. Indeed, such language-independent studies avoid having to define and investigate the same concept for different formalisms. In recent years, a promising framework for such language-independent investigations is the algebraic *approximation fixpoint theory* (AFT) (Denecker, Marek, and Truszczyński 2003), which conceives of KR-formalisms as operators over a lattice (such as the immediate consequence operator from logic programming). AFT can represent a wide variety of KR-formalisms; such as propositional logic programming (Denecker, Bruynooghe, and Vennekens 2012; Heyninck 2024b; Heyninck and Bogaerts 2023a), default logic (Denecker, Marek, and Truszczyński 2003), autoepistemic logic (Denecker, Marek, and Truszczyński 2003), abstract argumentation and abstract dialectical frameworks (Strass 2013), hybrid MKNF (Liu and You 2022) and SHACL (Bogaerts and Jakubowski 2021), and was shown to be fruitful for language-independent studies of concepts such as splitting (Vennekens, Gilis, and Denecker 2006), or non-determinism (Heyninck, Arieli, and Bogaerts 2022).

In this paper, we give an algebraic, operator-based account of conditional independence. In more detail, the paper makes the following contributions:
(1) Definition of conditional independence in an operator-based, algebraic framework, providing a notion applicabble to any formalism that admits an operator-based characterization, such as the ones mentioned above.
(2) Proof of crucial properties about conditional independence, including that search for fixpoints of an (approximation) operator over conditionally indepedent modules.
(3) Fixed-parameter tractability results based on conditional independence.
(4) A proof-of-concept application to normal logic programs under various semantics.
(5) Establish connections with existing work on conditional independence in KR.

Outline of the Paper: The necessary preliminaries on logic programming (Section 2.1), lattices (Section 2.2) and approximation fixpoint theory (Section 2.3) are introduced in Section 2. The concept of conditional independence of sublattices w.r.t. an operator is introduced and studied in Section 3, and applied to approximation operators in Section 4. Fixed-parameter complexity results are shown in Section 5. This theory is applied to the semantics of normal logic programs in Section 6. Related work is discussed in Section 7, after which the paper is concluded (Section 8).

2 Background and Preliminaries

In this section, we recall the necessary basics of logic programming, abstract algebra, and AFT. We refer to the literature for more detailed introductions on logic programming semantics based on four-valued operators (Fitting 1991; Denecker, Bruynooghe, and Vennekens 2012), order theory (Davey and Priestley 2002) and AFT (Bogaerts 2015).

2.1 Logic Programming

We assume a set of atoms $\mathcal{A}$ and a language $\mathcal{L}$ built up from atoms, conjunction $\wedge$ and negation not. A (propositional normal) logic program $\mathcal{P}$ (a nlp, for short) is a finite set of rules of the form $p \leftarrow p_1 \wedge \ldots \wedge p_n \wedge \mathrm{not}\, q_1 \wedge \cdots \wedge \mathrm{not}\, q_m$, where $p, p_1, \ldots, p_n, q_1, \ldots, q_m$ are atoms that may include the propositional constants $\top$ (representing truth) and $\bot$ (falsity). A rule is *positive* if there are no negations in the rule's bodies, and a program is positive if so are all its rules. We use the four-valued bilattice consisting of truth values U, F, T and C ordered by $\leq_i$ by: $\mathsf{U} \leq_i \mathsf{F} \leq_i \mathsf{C}$ and $\mathsf{U} \leq_i \mathsf{T} \leq_i \mathsf{C}$, and ordered by $\leq_t$ by: $\mathsf{F} \leq_t \mathsf{C} \leq_t \mathsf{T}$ and $\mathsf{F} \leq_t \mathsf{U} \leq_t \mathsf{T}$. We also assume a $\leq_t$-involution $-$ on $\leq_t$ (i.e, $-\mathsf{F} = \mathsf{T}$, $-\mathsf{T} = \mathsf{F}$, $-\mathsf{U} = \mathsf{U}$ and $-\mathsf{C} = \mathsf{C}$). A *four-valued interpretation* of a program $\mathcal{P}$ is a pair (x, y), where $x \subseteq \mathcal{A}_\mathcal{P}$ is the set of the atoms that are assigned a value in $\{\mathsf{T}, \mathsf{C}\}$ and $y \subseteq \mathcal{A}_\mathcal{P}$ is the set of atoms assigned a value in $\{\mathsf{T}, \mathsf{U}\}$. Somewhat skipping ahead to section 2.2, the intuition here is that x (y) is a lower (upper) approximation of the true atoms. Interpretations are compared by the *information order* $\leq_i$, where $(x, y) \leq_i (w, z)$ iff $x \subseteq w$ and $z \subseteq y$ (sometimes called "precision" order), and by the *truth order* $\leq_t$, where $(x, y) \leq_t (w, z)$ iff $x \subseteq w$ and $y \subseteq z$ (increased 'positive' evaluations). Truth assignments to complex formulas are then recursively defined as follows:

- $(x, y)(p) = \begin{cases} \mathsf{T} & \text{if } p \in x \text{ and } p \in y, \\ \mathsf{U} & \text{if } p \notin x \text{ and } p \in y, \\ \mathsf{F} & \text{if } p \notin x \text{ and } p \notin y, \\ \mathsf{C} & \text{if } p \in x \text{ and } p \notin y. \end{cases}$
- $(x, y)(\mathrm{not}\ \phi) = -(x, y)(\phi)$,
- $(x, y)(\psi \wedge \phi) = \bigsqcup_{\leq_t} \{(x, y)(\phi), (x, y)(\psi)\}$,

A four-valued interpretation of the form (x, x) may be associated with a *two-valued* (or *total*) interpretation x, in which for an atom p, $x(p) = \mathsf{T}$ if $p \in x$ and $x(p) = \mathsf{F}$ otherwise. We say that (x, y) is a *three-valued* (or *consistent*) interpretation, if $x \subseteq y$. Note that in consistent interpretations there are no C-assignments.

We now consider the two- and four-valued immediate consequence operators for nlps, defined as follows:

Definition 1. *Given a nlp $\mathcal{P}$ and a two-valued interpretation x, we define:*

$$IC_\mathcal{P}(x) = \{p \in \mathcal{A}_\mathcal{P} \mid p \leftarrow \psi \in \mathcal{P}, (x, x)(\psi) = \mathsf{T}\}.$$

For a four-valued interpretation (x, y), we define:

$$\mathcal{IC}^l_\mathcal{P}(x, y) = \{p \mid p \leftarrow \psi \in \mathcal{P}, (x, y)(\psi) \in \{\mathsf{T}, \mathsf{C}\}\},$$
$$\mathcal{IC}^u_\mathcal{P}(x, y) = \{p \mid p \leftarrow \psi \in \mathcal{P}, (x, y)(\psi) \in \{\mathsf{U}, \mathsf{T}\}\},$$
$$\mathcal{IC}_\mathcal{P}(x, y) = (\mathcal{IC}^l_\mathcal{P}(x, y), \mathcal{IC}^u_\mathcal{P}(x, y)).$$

Again somewhat skipping ahead to Section 2.2, denoting by $2^\mathcal{A}$ the powerset of $\mathcal{A}$, $IC_\mathcal{P}$ is an operator on the lattice $\langle 2^\mathcal{A}, \subseteq\rangle$ that derives all heads of rules with true bodies. $\mathcal{IC}_\mathcal{P}$, on the other hand, is a generalisation of this operator to $(2^\mathcal{A})^2$.

2.2 Lattices and Sub-Lattices

We recall some necessary preliminaries on set theory and (sub-)lattices. A lattice is a partially ordered set $L = \langle \mathcal{L}, \leq\rangle$ where every two elements $x, y \in \mathcal{L}$ have a least upper $x \sqcup y$ and a greatest lower bound $x \sqcap y$. We will often also refer to a lattice $\langle \mathcal{L}, \leq\rangle$ by its set of elements $\mathcal{L}$. A lattice is complete if every set $X \subseteq \mathcal{L}$ has a least upper (denoted $\bigsqcup X$) and a greatest lower bound (denoted $\bigsqcap X$). $\langle 2^\mathcal{A}, \subseteq\rangle$ is an example of a complete lattice. x is a fixpoint of O if $x = O(x)$, and the least fixpoint of O is denoted $\mathrm{lfp}(O)$.

We now provide background on how to (de)compose lattices into sub-lattices, following Vennekens, Gilis, and Denecker (2006). Let I be a set, which we call the *index set*, and for each $i \in I$, let $\mathcal{L}_i$ be a set. The product set $\bigotimes_{i\in I} \mathcal{L}_i$ is the following set of functions:

$$\bigotimes_{i\in I} \mathcal{L}_i = \{f \mid f : I \to \bigcup_{i\in I} \mathcal{L}_i \text{ s.t. } \forall i \in I :\ f(i) \in \mathcal{L}_i\}.$$

The product set $\bigotimes_{i\in I} \mathcal{L}_i$ contains all ways of selecting one element of every set $\mathcal{L}_i$. E.g. for the sets $\mathcal{L}_1 = \{\emptyset, \{p\}\}$ and $\mathcal{L}_2 = \{\emptyset, \{q\}\}$, $\bigotimes_{i\in\{1,2\}} \mathcal{L}_i$ contains, among others, f and f' with $f(1) = f(2) = \emptyset$ and $f'(1) = \emptyset$ and $f'(2) = \{q\}$. For finite $I = \{1, \ldots, n\}$, the product $\bigotimes_{i\in I} \mathcal{L}_i$ is (isomorphic to) the cartesian product $\mathcal{L}_1 \times \ldots \times \mathcal{L}_n$. We will also denote $\bigotimes_{i\in\{1,2\}} \mathcal{L}_i$ by $\mathcal{L}_1 \otimes \mathcal{L}_2$ to avoid clutter.

If each $\mathcal{L}_j$ is partially ordered by some $\leq_j$, this induces the product order $\leq_\otimes$ on $\bigotimes_{j\in I} \mathcal{L}_j$: for all $x, y \in \bigotimes_{j\in I} \mathcal{L}_j$, $x \leq_\otimes y$ iff for all $j \in I$, $x(j) \leq_j y(j)$. We will sometimes denote the product order over $\bigotimes_{j\in I} \mathcal{L}_j$ by $\leq^I_\otimes$. It can be easily shown that if all $\langle \mathcal{L}_j, \leq_j\rangle$ are (complete) lattices, then $\langle \bigotimes_{j\in I} \mathcal{L}_j, \leq_\otimes\rangle$ is also a (complete) lattice, called the *product lattice* of the lattices $\mathcal{L}_j$.

We denote, for $x \in \bigotimes_{i\in I} \mathcal{L}_i$ and $i \in I$, $x_{|i} \in \mathcal{L}_i$ as $f(i)$, and for $J \subseteq I$ we denote $x_{|J}$ by $\bigotimes_{i\in J} x_i$. For example, using $\mathcal{L}_1$ and $\mathcal{L}_2$ as in the example above, $\emptyset \times \{q\}_{|1} = \emptyset$. Likewise, we denote by $x_i \otimes x_j$ the element $x \in \mathcal{L}_i \otimes \mathcal{L}_j$ s.t. $x_{|k} = x_k$ for $k = i, j$, and we lift this to sets as usual.

2.3 Approximation Fixpoint Theory

We recall basic notions from approximation fixpoint theory (AFT) by (Denecker, Marek, and Truszczyński 2000).

Given a lattice $L = \langle \mathcal{L}, \leq\rangle$, we let $L^2 = \langle \mathcal{L}^2, \leq_i, \leq_t\rangle$ be the structure (called *bilattice*), in which $\mathcal{L}^2 = \mathcal{L} \times \mathcal{L}$, and for every $x_1, y_1, x_2, y_2 \in \mathcal{L}$,

- $(x_1, y_1) \leq_i (x_2, y_2)$ if $x_1 \leq x_2$ and $y_1 \geq y_2$,
- $(x_1, y_1) \leq_t (x_2, y_2)$ if $x_1 \leq x_2$ and $y_1 \leq y_2$.

An *approximating operator* $\mathcal{O} : \mathcal{L}^2 \to \mathcal{L}^2$ of an operator $O : \mathcal{L} \to \mathcal{L}$ is an operator that maps every approximation (x, y) of an element z to an approximation (x', y') of another element $O(z)$, thus approximating the behavior of the

approximated operator O. As an example, $\mathcal{IC}_\mathcal{P}$ is an approximation operator of $IC_\mathcal{P}$.

Definition 2. *Let $O_\mathcal{L} : \mathcal{L} \to \mathcal{L}$ and $\mathcal{O} : \mathcal{L}^2 \to \mathcal{L}^2$. (1) $\mathcal{O}$ is* $\leq_i$-monotonic*, if when $(x_1, y_1) \leq_i (x_2, y_2)$, also $\mathcal{O}(x_1, y_1) \leq_i \mathcal{O}(x_2, y_2)$; (2) $\mathcal{O}$ is* approximating*, if it is $\leq_i$-monotonic and for any $x \in \mathcal{L}$, $(\mathcal{O}(x, x))_1 = (\mathcal{O}(x, x))_2$;*[1] *(3) $\mathcal{O}$ is an* approximation *of $O_\mathcal{L}$, if it is $\leq_i$-monotonic and $\mathcal{O}$* extends *O, that is: $(\mathcal{O}(x, x))_1 = (\mathcal{O}(x, x))_2 = O_\mathcal{L}(x)$.*

To avoid clutter, we denote $(\mathcal{O}(x, y))_1$ by $\mathcal{O}_l(x, y)$ and $(\mathcal{O}(x, y))_2$ by $\mathcal{O}_u(x, y)$ (for *lower* and *upper* bound).

The *stable operator*, defined next, is used for expressing the semantics of many non-monotonic formalisms.

Definition 3. *Given a complete lattice $L = \langle \mathcal{L}, \leq \rangle$, let $\mathcal{O} : \mathcal{L}^2 \to \mathcal{L}^2$ be an approximating operator. $\mathcal{O}_l(\cdot, y) = \lambda x.\mathcal{O}_l(x, y)$, i.e.: $\mathcal{O}_l(\cdot, y)(x) = \mathcal{O}_l(x, y)$ (and similarly for the upper bound operator $\mathcal{O}_u$). The* stable operator for $\mathcal{O}$ *is: $S(\mathcal{O})(x, y) = (lfp(\mathcal{O}_l(., y)), lfp(\mathcal{O}_u(x, .))$.*

The components $lfp(\mathcal{O}_l(., y))$ and $lfp(\mathcal{O}_u(x, .)$ of $S(\mathcal{O})$ will be denoted by $C(\mathcal{O}_l)(y)$ respectively $C(\mathcal{O}_u)(x)$.

Stable operators capture the idea of minimizing truth, since for any $\leq_i$-monotonic operator $\mathcal{O}$ on $\mathcal{L}^2$, fixpoints of the stable operator $S(\mathcal{O})$ are $\leq_t$-minimal fixpoints of $\mathcal{O}$ (Denecker, Marek, and Truszczyński 2000, Theorem 4). Altogether, we obtain the following notions: Given a complete lattice $L = \langle \mathcal{L}, \leq \rangle$, let $\mathcal{O} : \mathcal{L}^2 \to \mathcal{L}^2$ be an approximating operator, (x, y) is

- a *Kripke-Kleene fixpoint* of $\mathcal{O}$ if $(x, y) = \mathrm{lfp}_{\leq_i}(\mathcal{O}(x, y))$;
- a *three-valued stable* fixpoint of $\mathcal{O}$ if $(x, y) = S(\mathcal{O})(x, y)$;
- a *two-valued stable fixpoints* of $\mathcal{O}$ if $(x, y) = S(\mathcal{O})(x, y)$; and $x = y$;
- the *well-founded fixpoint* of $\mathcal{O}$ if it is the $\leq_i$-minimal (three-valued) stable model fixpoint of $\mathcal{O}$.

It has been shown that every approximation operator admits a unique $\leq_i$-minimal stable fixpoint (Denecker, Marek, and Truszczyński 2000). Pelov, Denecker, and Bruynooghe (2007) show that for normal logic programs, the fixpoints based on the four-valued immediate consequence operator $\mathcal{IC}_\mathcal{P}$ (recall Definition 1) for a logic program $\mathcal{P}$ give rise to the following correspondences: the three-valued stable fixpoints of $\mathcal{IC}_\mathcal{P}$ coincide with the three-valued stable semantics as defined by Przymusinski (1990), the well-founded fixpoint of of $\mathcal{IC}_\mathcal{P}$ coincides with the homonymous semantics (Przymusinski 1990; Van Gelder, Ross, and Schlipf 1991), and the two-valued stable fixpoints of $\mathcal{IC}_\mathcal{P}$ coincide with the two-valued (or total) stable models.

[1] In some papers (Denecker, Marek, and Truszczyński 2000), an approximation operator is defined as a symmetric $\leq_i$-monotonic operator, i.e. a $\leq_i$-monotonic operator s.t. for every $x, y \in \mathcal{L}$, $\mathcal{O}(x, y) = (\mathcal{O}_l(x, y), \mathcal{O}_l(y, x))$ for some $\mathcal{O}_l : \mathcal{L}^2 \to \mathcal{L}$. However, the weaker condition we take here (taken from Denecker, Marek, and Truszczynski (2002)) is actually sufficient for most results.

3 Conditional Independence

Conditional independence in an operator-based setting is meant to formalize the idea that for the application of an operator to a lattice consisting of three sub-lattices $\mathcal{L}_1$, $\mathcal{L}_2$ and $\mathcal{L}_3$, full information about $\mathcal{L}_3$ allows us to ignore $\mathcal{L}_2$ when applying O to $\mathcal{L}_1 \otimes \mathcal{L}_3$. In more detail, it means that the operator O over $\bigotimes_{i\in\{1,2,3\}} \mathcal{L}_i$ can be decomposed in two operators $O_{1,3}$ and $O_{2,3}$ over the sub-lattices $\mathcal{L}_1 \otimes \mathcal{L}_3$ respectively $\mathcal{L}_2 \otimes \mathcal{L}_3$.

Definition 4. *Let O be an operator $\bigotimes_{i\in\{1,2,3\}} \mathcal{L}_i$. The lattices $\mathcal{L}_1$ and $\mathcal{L}_2$* are independent w.r.t. *$\mathcal{L}_3$ according to O, in symbols: $\mathcal{L}_1 \perp\!\!\!\perp_O \mathcal{L}_2 \mid \mathcal{L}_3$, if there exist operators*

$$O_{i,3} : \mathcal{L}_i \otimes \mathcal{L}_3 \to \mathcal{L}_i \otimes \mathcal{L}_3 \quad \text{for } i = 1, 2$$

s.t. for every $x_1 \in \mathcal{L}_1, x_2 \in \mathcal{L}_2, x_3 \in \mathcal{L}_3$ it holds that:

$$O(x_1 \otimes x_2 \otimes x_3)_{|1,3} = O_{1,3}(x_1 \otimes x_3) \quad \text{and}$$
$$O(x_1 \otimes x_2 \otimes x_3)_{|2,3} = O_{2,3}(x_2 \otimes x_3)$$

Thus, two sub-lattices $\mathcal{L}_1$ and $\mathcal{L}_2$ are independent w.r.t. $\mathcal{L}_3$ according to O if, once we have full information about $\mathcal{L}_3$, information about $\mathcal{L}_2$ does not contribute anything in the application of O when restricted to $\mathcal{L}_1$ (and vice versa). Where $\mathcal{L}_1 \perp\!\!\!\perp_O \mathcal{L}_2 \mid \mathcal{L}_3$, we will also call $\mathcal{L}_3$ the *conditional pivot*, and will refer to members of $\mathcal{L}_3$ as such as well.

Example 1. *Consider the logic program $\mathcal{P}$ using atoms for* `inf`*ected,* `vac`*cinated and* `cnct`*act:*

$$r_1 : \texttt{inf(b)} \leftarrow \texttt{inf(a)}, \texttt{cnct(a, b)}, \text{not } \texttt{vac(b)}.$$
$$r_2 : \texttt{inf(c)} \leftarrow \texttt{inf(a)}, \texttt{cnct(a, c)}, \text{not } \texttt{vac(c)}.$$
$$r_3 : \texttt{inf(a)}., \; r_4 : \texttt{cnct(a, b)}., \; r_5 : \texttt{cnct(a, c)}.$$

Notice that, as soon as we know that `inf(a)`*. is the case, we can decompose the search for models into two independent parts, as can also be seen in the dependency graph in figure 1.*

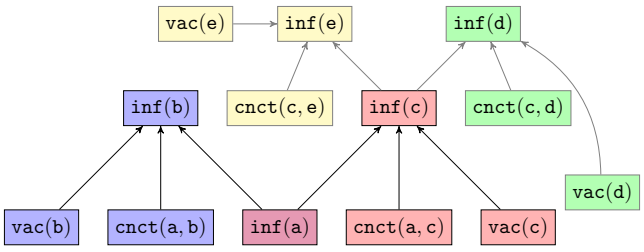


Figure 1: A dependency graph for the program $\mathcal{P}_1$ (Example 1), and its extension $\mathcal{P}_2$ (Example 2, atoms only occuring in $\mathcal{P}_2$ have gray outlines.).

As a product lattice consisting of power sets of sets $\mathcal{A}_1, \ldots, \mathcal{A}_3$ is isomorphic to the powerset of the union of these sets $\mathcal{A}_1 \cup \ldots \cup \mathcal{A}_3$, we shall use them interchangeably. We let:

$$\mathcal{A}_b = \{\texttt{inf(b)}, \texttt{cnct(a, b)}, \texttt{vac(b)}\}$$
$$\mathcal{A}_c = \{\texttt{inf(c)}, \texttt{cnct(a, c)}, \texttt{vac(c)}\}$$
$$\mathcal{A}_a = \{\texttt{inf(a)}\}$$

We see that $2^{\mathcal{A}_b} \perp\!\!\!\perp_{IC_\mathcal{P}} 2^{\mathcal{A}_c} \mid 2^{\mathcal{A}_a}$*, by observing that:*

$$IC_\mathcal{P}^{\mathcal{A}_b,\mathcal{A}_a} = IC_{\mathcal{P}^{\mathcal{A}_b,\mathcal{A}_a}} \text{ and } IC_\mathcal{P}^{\mathcal{A}_c,\mathcal{A}_a} = IC_{\mathcal{P}^{\mathcal{A}_c,\mathcal{A}_a}}$$

where $\mathcal{P}^{\mathcal{A}_b,\mathcal{A}_a} = \{r_1, r_3, r_4\}$ *and* $\mathcal{P}^{\mathcal{A}_c,\mathcal{A}_a} = \{r_2, r_3, r_5\}$*. It is easily verified that for every* $x_1 \cup x_2 \cup x_3 \subseteq \mathcal{A}_\mathcal{P}$*, it holds that* $IC_\mathcal{P}(x_b \cup x_c \cup x_a) \cap (\mathcal{A}_i \cup \mathcal{A}_a) = IC_{\mathcal{P}^{\mathcal{A}_i,\mathcal{A}_a}}(x_i \cup x_a)$ *for any* $i = b, c$*.*

Intuitively, our notion of conditional independence relates to the analogous notion known from probability theory as follows: two events x_1 and x_2 in probability theory are independent conditional on a third x_3 if we can calculate the probability of x_1 given x_2 and x_3 as the probability of x_1 given x_3, i.e. $P(x_1 \mid x_2x_3) = P(x_1 \mid x_3)$. The idea of conditional probability relative to an operator is the same: we can apply the operator to one of the two sub-lattices independently of the second, independent sub-lattice.

We now undertake a study of the properties of operators that respect conditional independencies. In probability theory, an equivalent definition of conditional independence is given by: $P(x_1, x_2|x_3) = P(x_1|x_3)P(x_2|x_3)$. This property or definition is mirrored in the following fact, where we show that the application of an operator over the entire lattice can likewise be split up over the two independent sub-lattices $\mathcal{L}_1$ and $\mathcal{L}_2$ (together with the conditional pivot $\mathcal{L}_3$):

Fact 1. [2] *Let an operator* O *on* $\otimes_{i\in\{1,2,3\}}\mathcal{L}_i$ *s.t.* $\mathcal{L}_1 \perp\!\!\!\perp_O \mathcal{L}_2 \mid \mathcal{L}_3$ *be given. Then for any* $x_1 \otimes x_2 \otimes x_3 \in \bigotimes_{i\in\{1,2,3\}} \mathcal{L}_i$*, it holds that:*

$$\begin{aligned} O(x_1 \otimes x_2 \otimes x_3) &= O_{1,3}(x_1 \otimes x_3) \otimes O_{2,3}(x_2 \otimes x_3)_{|2} \\ &= O_{1,3}(x_1 \otimes x_3)_{|1} \otimes O_{2,3}(x_2 \otimes x_3). \end{aligned}$$

Furthermore, for any $i, j = 1, 2$*,* $i \neq j$*,* $x_i \in \mathcal{L}_i, x_j, x'_j \in \mathcal{L}_j$ *and* $x_3 \in \mathcal{L}_3$ *it holds that:*

$$O(x_i \otimes x_j \otimes x_3)_{|i,3} = O(x_i \otimes x'_j \otimes x_3)_{|i,3}$$

The output of an operator w.r.t. the conditional pivot only depends on the input w.r.t. the conditional pivot:

Lemma 1. *Let an operator* O *on* $\otimes_{i\in\{1,2,3\}}\mathcal{L}_i$ *s.t.* $\mathcal{L}_1 \perp\!\!\!\perp_O \mathcal{L}_2 \mid \mathcal{L}_3$ *be given. Then* $O_{2,3}(x_2 \otimes x_3)_{|3} = O_{1,3}(x_1 \otimes x_3)_{|3}$ *for every* $x_1 \otimes x_2 \otimes x_3 \in \otimes_{i\in\{1,2,3\}}\mathcal{L}_i$*.*

We now show one of the central results, namely that fixpoints of an operator O respecting independence of $\mathcal{L}_1$ and $\mathcal{L}_2$ w.r.t. $\mathcal{L}_3$ can be obtained by combining the fixpoints of $O_{1,3}$ and $O_{2,3}$. Thus, the search for fixpoints, a central task in KR, can be split into two parallel, smaller searches:

Proposition 1. *Let an operator* O *on* $\bigotimes_{i\in\{1,2,3\}} \mathcal{L}_i$ *s.t.* $\mathcal{L}_1 \perp\!\!\!\perp_O \mathcal{L}_2 \mid \mathcal{L}_3$ *be given. Then* $x_1 \otimes x_2 \otimes x_3 = O(x_1 \otimes x_2 \otimes x_3)$ *iff* $x_1 \otimes x_3 = O_{1,3}(x_1 \otimes x_3)$ *and* $x_2 \otimes x_3 = O_{2,3}(x_2 \otimes x_3)$ *(for any* $x_1 \otimes x_2 \otimes x_3 \in \bigotimes_{i\in\{1,2,3\}} \mathcal{L}_i$*).*

A second central insight is that monotonicity is preserved when moving between a product lattice and its components:

Proposition 2. *Let an operator* O *on* $\bigotimes_{i\in\{1,2,3\}} \mathcal{L}_i$ *s.t.* $\mathcal{L}_1 \perp\!\!\!\perp_O \mathcal{L}_2 \mid \mathcal{L}_3$ *be given. Then* O *is* $\leq_\otimes$*-monotonic iff* $O_{i,3}$ *over* $\mathcal{L}_i \otimes \mathcal{L}_3$ *is* $\leq_\otimes^{i,3}$*-monotonic for* $i = 1, 2$*.*

[2] Proofs of all results are available in an extended version of this article (Heyninck 2024a).

Finally, for monotonic operators over complete lattices, the *least* fixed points can be obtained by combining the least fixed points of conditionally independent sub-lattices:

Proposition 3. *Let a* $\leq_\otimes$*-monotonic operator* O *on the complete lattice* $\bigotimes_{i\in\{1,2,3\}} \mathcal{L}_i$ *s.t.* $\mathcal{L}_1 \perp\!\!\!\perp_O \mathcal{L}_2 \mid \mathcal{L}_3$ *be given. Then* $x_1 \otimes x_2 \otimes x_3$ *is the least fixed point of* O *iff* $x_i \otimes x_3$ *is the least fixed point of* $O_{i,3}$ *(for* $i = 1, 2$ *and any* $x_1 \otimes x_2 \otimes x_3 \in \bigotimes_{i\in\{1,2,3\}} \mathcal{L}_i$*).*

4 Conditional Independence and AFT

The notion of conditional independence is immediately applicable to approximation operators. In this section, we derive results on the modularisation of AFT-based semantics based on the results from the previous section.

As observed by Vennekens, Gilis, and Denecker (2006), the bilattice $\mathcal{L}^2$ of a product lattice $\mathcal{L} = \bigotimes_{i\in I} \mathcal{L}_i$ is isomorphic to the product lattice of bilattices $\bigotimes_{i\in I} \mathcal{L}_i^2$, and we move between these two constructs without further ado.

As an approximation operator is a $\leq_i$-monotonic operator, we immediately obtain that the search for fixpoints, including the Kripke-Kleene fixpoints, can be split on the basis of conditional independence of an approximator:

Proposition 4. *Let an approximation operator* $\mathcal{O}$ *over a bilattice of the product lattice* $\bigotimes_{i\in\{1,2,3\}} \mathcal{L}_i$ *be given s.t.* $\mathcal{L}_1^2 \perp\!\!\!\perp_\mathcal{O} \mathcal{L}_2^2 \mid \mathcal{L}_3^2$*. Then the following hold:*

- (x, y) *is the Kripke-Kleene fixpoint of* $\mathcal{O}$ *iff* $(x_{|i,3}, y_{|i,3})$ *is the Kripke-Kleene fixpoint of* $\mathcal{O}_{i,3}$ *for* $i = 1, 2$*.*
- (x, y) *is a fixpoint of* $\mathcal{O}$ *iff* $(x_{|i,3}, y_{|i,3})$ *is a fixpoint of* $\mathcal{O}_{i,3}$ *for* $i = 1, 2$*.*

We now turn our attention to the stable operators. We first investigate the relation between an approximation operator and the lower and upper-bound component of this operator when it comes to respecting independencies. It turns out that the component operators $\mathcal{O}_l$ and $\mathcal{O}_u$ respect conditional independencies, and, vice-versa, that the respect of the two component operators of conditional independencies implies respect of these independencies by the approximator:

Proposition 5. *Let an approximation operator* $\mathcal{O}$ *over a bilattice of the product lattice* $\bigotimes_{i\in\{1,2,3\}} \mathcal{L}_i$ *be given. Then* $\mathcal{L}_1^2 \perp\!\!\!\perp_\mathcal{O} \mathcal{L}_2^2 \mid \mathcal{L}_3^2$ *iff* $\mathcal{L}_1^2 \perp\!\!\!\perp_{\mathcal{O}_l} \mathcal{L}_2^2 \mid \mathcal{L}_3^2$ *and* $\mathcal{L}_1^2 \perp\!\!\!\perp_{\mathcal{O}_u} \mathcal{L}_2^2 \mid \mathcal{L}_3^2$*.* [3]

Stable operators respect the conditional independencies respected by the approximator from which they derive:

Proposition 6. *Let an approximation operator* $\mathcal{O}$ *over a bilattice of the complete product lattice* $\bigotimes_{i\in\{1,2,3\}} \mathcal{L}_i$ *be given s.t.* $\mathcal{L}_1^2 \perp\!\!\!\perp_\mathcal{O} \mathcal{L}_2^2 \mid \mathcal{L}_3^2$*. Then* $\mathcal{L}_1 \perp\!\!\!\perp_{\mathcal{C}(\mathcal{O}_l)} \mathcal{L}_2 \mid \mathcal{L}_3$ *and* $\mathcal{L}_1 \perp\!\!\!\perp_{\mathcal{C}(\mathcal{O}_u)} \mathcal{L}_2 \mid \mathcal{L}_3$*.*

This allows us to derive another central result, stating that search for stable fixpoints, including the well-founded one, can be split up on the basis of conditional independencies.

[3] Notice that we slightly abuse notation here, as $\mathcal{O}_l$ and $\mathcal{O}_u$ map from $\mathcal{L}^2$ to $\mathcal{L}$. However, one can easily obtain an operator $\mathcal{O}'_l$ on $\mathcal{L}^2$ by e.g. defining $\mathcal{O}_l(x, y) = (\mathcal{O}'_l(x, y), \top)$.

Proposition 7. *Let an approximation operator $\mathcal{O}$ over a bilattice of the product lattice $\bigotimes_{i\in\{1,2,3\}} S_i$ be given s.t. $\mathcal{L}_1^2 \perp\!\!\!\perp_{\mathcal{O}} \mathcal{L}_2^2 \mid \mathcal{L}_3^2$. Then:*

1. *(x, y) is a fixpoint of $S(\mathcal{O})$ iff $(x_{|i,3}, y_{|i,3})$ is a fixpoint of $S(\mathcal{O}_{i,3})$ for $i = 1, 2$.*
2. *(x, y) is the well-founded fixpoint of $\mathcal{O}$ iff $(x_{|i,3}, y_{|i,3})$ is the well-founded fixpoint of $\mathcal{O}_{i,3}$ for $i = 1, 2$.*

5 Parametrized Complexity Results Based on Conditional Independence

In this section, we show how the notion of conditional independence can be made useful to break up reasoning tasks into smaller tasks that can be solved in parallel. We do this by introducing *conditional independence trees*, and illustrate its usefulness by showing that the size of the induced modules serves as a parameter for the fixed parameter tractability of calculating the well-founded fixpoint.

5.1 Preliminaries on Parametrized Complexity

In this section, we recall the necessary background on (parametrized) complexity. For more detailed references, we refer to (Downey and Fellows 2013). We assume familiarity with basics on the polynomial hierarchy. An instance of a *parametrized problem* L is a pair $(I, k) \in \Sigma^* \times \mathbb{N}$ for some finite alphabet Σ, and we call I the *main part* and k the *parameter*. Where $|I|$ denotes the cardinality of I, L is *fixed-parameter tractable* if there exists a computable function f and a constant c such that $(I, k) \in L$ is decidable in time $O(f(k)|I|^c)$. Such an algorithm is called a *fixed-parameter tractable algorithm*. Thus, the intuition is that when $f(k)$ remains sufficiently small, the problem remains tractable no matter the size of I.

5.2 Conditional Independence Trees

How can conditional independence be used to make reasoning more efficient? Conditional independence allows to split up the application of an operator over lattice elements to parallel reasoning over elements of the sub-lattices.These sub-lattices thus define *sub-modules* for parallel reasoning. Conditional independence only allows us to split up reasoning into two modules, thus only splitting the search space in half at best. However, the operator allowing, these modules can again be split up into smaller modules, leading to a tree structure of nested modules, which we call *conditional independence trees*:

Definition 5. *Let $\bigotimes_{i\in I} \mathcal{L}_i$ be a product lattice. We call a binary labelled tree (V, E, ν) a* conditional independence tree *for $\mathcal{O}$ (in short, CIT) if the following holds:*

- $\nu : V \to 2^I \times 2^I \times 2^I$,
- *the root is labelled $\langle I_1, I_2, I_3\rangle$ where I_1, I_2, I_3 is a partition of I,*
- *for every $(v_1, v_2), (v_1, v_3) \in E$, where $\nu(v_i) = \langle I_1^i, I_2^i, I_3^i\rangle$ for $i = 1, 2, 3$, $I_j^1 \cup I_3^1 = I_1^{j+1} \cup I_2^{j+1} \cup I_3^{j+1}$ for $j = 1, 2$.*
- *$\bigotimes_{i\in I_1} \mathcal{L}_i \perp\!\!\!\perp_{O_{I_1\cup I_2\cup I_3}} \bigotimes_{i\in I_2} \mathcal{L}_i \mid \bigotimes_{i\in I_3} \mathcal{L}_i$ for every $v \in V$ with $\nu(v) = \langle I_1, I_2, I_3\rangle$.*

Thus, a CIT is a tree where each vertex is labelled with a partition $\langle I_1, I_2, I_3\rangle$ of a sub-lattice $\bigotimes_{i\in I_1\cup I_2\cup I_3} \mathcal{L}_i$ s.t. $\bigotimes_{i\in I_1} \mathcal{L}_i$ is independent w.r.t. $\bigotimes_{i\in I_2} \mathcal{L}_i$ given $\bigotimes_{i\in I_3} \mathcal{L}_i$ according to O, and such that the labels of the leaves of a node compose the sub-lattices mentioned in the the labels of the parent node. Finally, the root of the tree should be a decomposition of the original lattice.

Example 2. *We consider an extension of Example 1 where $\mathcal{P}_2 = \{r_1, r_2, r_3, r_4, r_5, r_6, r_7, r_8\}$ with:*

$$r_6 : \texttt{inf(d)} \leftarrow \texttt{inf(c)}, \texttt{cnct(c,d)}, \text{not } \texttt{vac(d)}.$$
$$r_7 : \texttt{inf(e)} \leftarrow \texttt{inf(c)}, \texttt{cnct(c,e)}, \text{not } \texttt{vac(e)}.$$
$$r_8 : \texttt{cnct(c,d)}., \; r_9 : \texttt{cnct(c,e)}.$$

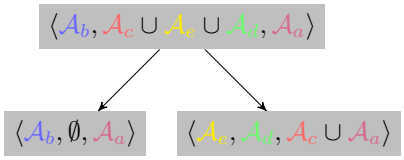
Furthermore, we let:

$$\mathcal{A}_e = \{\texttt{inf(e)}, \texttt{cnct(c,e)}, \texttt{vac(e)}\}$$
$$\mathcal{A}_d = \{\texttt{inf(d)}, \texttt{cnct(d,e)}, \texttt{vac(d)}\}$$

The dependency-graph for this program is given in Figure 1. We observe the following independencies:

$$2^{\mathcal{A}_b} \perp_{IC_{\mathcal{P}_2}} 2^{\mathcal{A}_c\cup\mathcal{A}_e\cup\mathcal{A}_d} \mid 2^{\mathcal{A}_a}$$
$$2^{\mathcal{A}_e} \perp_{IC_{\mathcal{P}_2}} 2^{\mathcal{A}_d} \mid 2^{\mathcal{A}_a\cup\mathcal{A}_c}$$

We can accordingly obtain the following CIT:



This CIT can be used to reduce global reasoning to parallel instances of local reasoning along the different components occuring in the leaves of the CIT. For example, we can calculate $\mathsf{WF}(\mathcal{P}_2) = \mathsf{WF}(\mathcal{P}_b\cup\mathcal{P}_a)\cup\mathsf{WF}(\mathcal{P}_e\cup\mathcal{P}_e\cup\mathcal{P}_c)\cup\mathsf{WF}(\mathcal{P}_d\cup\mathcal{P}_a\cup\mathcal{P}_a)$ *(where $\mathcal{P}_d = \{r_6, r_8\}$ and $\mathcal{P}_e = \{r_7, r_9\}$):*

$$\mathsf{WF}(\mathcal{P}_b \cup \mathcal{P}_a) = \{\texttt{inf(a)}, \texttt{cnct(a,b)}, \texttt{inf(b)}\}$$

$$\mathsf{WF}(\mathcal{P}_e \cup \mathcal{P}_c \cup \mathcal{P}_a) =$$
$$\{\texttt{inf(a)}, \texttt{cnct(a,c)}, \texttt{inf(c)}, \texttt{cnct(c,e)}, \texttt{inf(e)}\}$$

$$\mathsf{WF}(\mathcal{P}_d \cup \mathcal{P}_c \cup \mathcal{P}_a) =$$
$$\{\texttt{inf(a)}, \texttt{cnct(a,c)}, \texttt{inf(c)}, \texttt{cnct(c,d)}, \texttt{inf(d)}\}$$

$$\mathsf{WF}(\mathcal{P}_b \cup \mathcal{P}_c \cup \mathcal{P}_a \cup \mathcal{P}_e \cup \mathcal{P}_d) =$$
$$\mathsf{WF}(\mathcal{P}_b \cup \mathcal{P}_a) \cup \mathsf{WF}(\mathcal{P}_e \cup \mathcal{P}_e \cup \mathcal{P}_c) \cup \mathsf{WF}(\mathcal{P}_d \cup \mathcal{P}_c \cup \mathcal{P}_a)$$
$$= \{\texttt{inf(a)}, \texttt{cnct(a,b)}, \texttt{inf(b)}, \texttt{cnct(a,c)},$$
$$\texttt{inf(c)}, \texttt{cnct(c,e)}, \texttt{inf(e)}, \texttt{cnct(c,d)}, \texttt{inf(d)}\}$$

The following result shows that a CIT-tree correctly decomposes reasoning problems in KR:

Proposition 8. *Let an operator O over $\bigotimes_{i\in I} \mathcal{L}_i$ and CIT $T = (V, E, \nu)$ be given with V_l the leafs of T. Then:*

1. $\text{lfp}(O) = \bigotimes_{\langle I_1,I_2,I_3\rangle\in V_l} \text{lfp}(O_{I_1\cup I_3}) \otimes \text{lfp}(O_{I_2\cup I_3})$,
2. *x is a fixpoint of O iff for every $\langle I_1, I_2, I_3\rangle \in V_l$ and $i = 1, 2$, $x_{|I_i\cup I_3}$ is a fixpoint of $O_{I_i\cup I_3}$.*

We show how the decomposition of reasoning according to a CIT results in FPT-results, making the following

Important Assumption 1. *In the rest of this section, we assume that L is a powerset-lattice, i.e. $L = \langle 2^S, \subseteq \rangle$ for some set S, to ensure that cardinality used in Def. 6 is defined.*

Lifting this assumption to other lattices is straightforward, but requires one to define a notion of size on the lattice elements. We leave this for future work. We furthermore notice that we will use $|S|$ as the input size of the problems studied below. Notice that we do *not* consider $|2^S|$ as the input size, as this would make most results below trivial.

CIT-size encodes the size of modules and serving as basis for the parameter in the FPT-results below.

Definition 6. *Let an operator O over $\bigotimes_{i\in I} \mathcal{L}_i$, a CIT $T = (V, E, \nu)$ for O and V_I the leafs of T be given. The* CIT-partition-size *of O relative to (V, E, ν) is defined as* $\mathsf{CPS}(T) =$

$$\max(\{|\bigotimes_{i\in I_j} \mathcal{L}_i \otimes \bigotimes_{i\in I_3} \mathcal{L}_i| \mid \quad v \in V_\mathsf{I}, \nu(v) = \langle I_1, I_2, I_3\rangle, \; j = 1, 2\}).$$

The CIT-size *of O relative to T is defined as* $\mathsf{CS}(V, E, \nu) = \max(\mathsf{CPS}(V, E, \nu), |V_\mathsf{I}|)$.

Example 3. *In the CIT of Example 2, the CIT-size is equal to the size of the largest partition, namely $2^{|\mathcal{A}_e \cup \mathcal{A}_c \cup \mathcal{A}_a|} = 2^{|\mathcal{A}_d \cup \mathcal{A}_c \cup \mathcal{A}_a|} = 2^7$.*

We first show that computing the well-founded fixpoint is FPT with the CIT-partition-size as a parameter. We do this under the assumption that applying an operator O can be reduced by a call to an NP-oracle. This covers several interesting cases, e.g. ADFs (Strass and Wallner 2015) and the ultimate operator (Denecker, Marek, and Truszczyński 2004) for normal logic programs (see also Section 6).

Proposition 9. *Let a $\leq_\otimes$-monotonic operator O over the powerset lattice 2^S and a CIT T for O with CIT-size c be given. Assume that $O(x)$ can be computed by a call to an* NP*-oracle for any $x \subseteq S$. The least fixpoint of O can be computed in time $O(f(c).|S|)$.*

We now show similar results for skeptical and credulous inference, under the assumption that $O(x)$ is polynomial.

Proposition 10. *Let a $\leq_\otimes$-monotonic operator O over the powerset lattice 2^S, a CIT T for O with CIT-size c and some $\alpha \in S$ be given. Assume that $O(x)$ can be computed in polynomial time for any $x \subseteq S$:*

1. *Determining whether there exists some fixpoint x of O s.t. $\alpha \in x$ can be done in time $O(f(c).|S|)$.*
2. *Determining whether for every fixpoint x of O it holds that $\alpha \in x$ can be done in time $O(f(c).|S|)$.*

It should be noticed that we do not say anything here about the complexity of determining the CIT-partition-size. This is because, firstly, the generality of the operator-based framework makes it hard to make concrete claims about this very formalism-dependent question. Secondly, for concrete operators, efficiently looking for conditional independencies is a research topic on itself, left for future work.

6 Application to Logic Programs

In this section, we illustrate the theory developed in the previous section by applying it to normal logic programs. We can avoid clutter with a slight abuse of notation by writing $\mathcal{A}_1 \perp\!\!\!\perp_\mathcal{P} \mathcal{A}_2 \mid \mathcal{A}_3$ to denote $2^{\mathcal{A}_1} \perp\!\!\!\perp_{\mathcal{IC}_\mathcal{P}} 2^{\mathcal{A}_2} \mid 2^{\mathcal{A}_3}$.

We first define what we call the *marginalisation* of a program w.r.t. a set of atoms, which represents the syntactic counterpart of the modularised operator $O_{i,3}$.

Definition 7. *Let an nlp $\mathcal{P}$ and some $\mathcal{A} \subseteq \mathcal{A}_\mathcal{P}$ be given. We define $\mathcal{P}_\mathcal{A}$ as the program obtained by replacing in every rule $r \in \mathcal{P}$ every occurrence of an atom $p \in \mathcal{A}$ by $\bot$.*

For example, $\{p \leftarrow q, r, \text{not } s\}_{\{r,s\}} = \{p \leftarrow q, \bot, \top\}$. This transformation is correct, i.e. given a program $\mathcal{P}$ with $\mathcal{A}_1 \perp\!\!\!\perp_\mathcal{P} \mathcal{A}_2 \mid \mathcal{A}_3$, the marginalisation $\mathcal{P}_{\mathcal{A}_2}$ induces the $\mathcal{IC}$-operator for the sublattice $\mathcal{A}_1 \cup \mathcal{A}_3$:

Proposition 11. *Let a nlp $\mathcal{P}$ be given for which $\mathcal{A}_\mathcal{P}$ is partitioned into $\mathcal{A}_1 \cup \mathcal{A}_2 \cup \mathcal{A}_3$ s.t. $\mathcal{A}_1 \perp\!\!\!\perp_\mathcal{P} \mathcal{A}_2 \mid \mathcal{A}_3$. Then $\mathcal{IC}^{i,3}_\mathcal{P} = \mathcal{IC}_{\mathcal{P}_{\mathcal{A}_j}}$ (for $i, j = 1, 2$, $i \neq j$).*

We first observe that the search for supported, (partial) stable and well-founded models can be split up along conditionally independent sub-alphabets:

Corollary 1. *Let an nlp $\mathcal{P}$ be given for which $\mathcal{A}_\mathcal{P}$ is partitioned into $\mathcal{A}_1 \cup \mathcal{A}_2 \cup \mathcal{A}_3$ s.t. $\mathcal{A}_1 \perp\!\!\!\perp_\mathcal{P} \mathcal{A}_2 \mid \mathcal{A}_3$. $x_1 \cup x_2 \cup x_3$ is a supported (respectively three-valued stable) model of $\mathcal{P}$ iff $x_i \cup x_3$ is a supported (respectively three-valued stable) model of $\mathcal{P}_{\mathcal{A}_j}$ (for $i, j = 1, 2$ and $i \neq j$). The well-founded model of $\mathcal{P}$ can be obtained as $(x_1 \cup x_2 \cup x_3, y_1 \cup y_2 \cup y_3)$, where $(x_i \cup x_3, y_i \cup y_3)$ is the well-founded model of $\mathcal{P}_{\mathcal{A}_j}$ (for $i, j = 1, 2$, $i \neq j$).*

It is well-known that computing the well-founded fixpoint for the operator $\mathcal{IC}_\mathcal{P}$ can be done in polynomial time. The well-founded fixpoint for the ultimate operator Denecker, Marek, and Truszczynski (2002), which is NP-hard, is FPT for the CIT-size (shown in the extended version (Heyninck 2024a) as a corollary of Proposition 9). Likewise, Proposition 10 can be used to show a similar result for credulous and skeptical entailment under the stable model semantics.

We now make some observations on how to detect conditional independencies in a logic program based on its syntax. We first need some further preliminaries. The *dependency order* for a logic program $\mathcal{P}$, $\leq^\mathcal{P}_\text{dep} \subseteq \mathcal{A}_\mathcal{P} \times \mathcal{A}_\mathcal{P}$, is defined as $p \leq^\mathcal{P}_\text{dep} q$ iff there is some $r \in \mathcal{P}$ where q is the head of r and p occurs in the body of r. The *dependency graph*, denoted $\text{DP}(\mathcal{P})$ of $\mathcal{P}$ is the Hasse diagram of $\leq^\mathcal{P}_\text{dep}$. Figure 1 is an example of the (inverse of) a dependency graph.

A first conjecture could be that, a sufficient criterion fo $\mathcal{A}_1 \perp\!\!\!\perp_\mathcal{P} \mathcal{A}_2 \mid \mathcal{A}_3$ is that $\mathcal{A}_3$ graphically separates $\mathcal{A}_1$ and $\mathcal{A}_2$, i.e. given $\text{DP}(\mathcal{P}) = \langle \mathcal{A}_\mathcal{P}, V\rangle$, a set $\mathcal{A}_3$ s.t. $\langle \mathcal{A}_\mathcal{P} \setminus \mathcal{A}_3, V \cap ((\mathcal{A}_\mathcal{P} \setminus (\mathcal{A}_3) \times (\mathcal{A}_\mathcal{P} \setminus (\mathcal{A}_3))$ consists of two disconnected subgraphs $\langle \mathcal{A}_1, V \cap (\mathcal{A}_1 \times \mathcal{A}_1)\rangle$ and $\langle \mathcal{A}_2, V \cap (\mathcal{A}_2 \times \mathcal{A}_2)\rangle$ induces the conditional independence $\mathcal{A}_1 \perp\!\!\!\perp_\mathcal{P} \mathcal{A}_2 | \mathcal{A}_3$. However, this conjecture is too naive:

Example 4. *Consider the program $\mathcal{P} = \{a_1 \leftarrow \text{not} b_1; b_1 \leftarrow \text{not} a_1; e \leftarrow b_1\}$ and $\mathcal{P}_2 = \{a_2 \leftarrow \text{not} b_2; b_2 \leftarrow \text{not} a_2; e \leftarrow b_2\}$. This program has the following dependency graph:*

$$a_1 \longleftrightarrow b_1 \longrightarrow e \longleftarrow b_2 \longleftrightarrow a_2$$

We could conjecture $\{a_1, b_1\} \perp\!\!\!\perp_{\mathcal{P}} \{a_2, b_2\} | \{e\}$*, but this does not hold, as*

$$\begin{aligned} IC_{\mathcal{P}}(\{a_1, b_2\})_{|\{a_1,b_1,e\}} &= \{a_1, e\} \\ \neq IC_{\mathcal{P}}(\{a_1\})_{|\{a_1,b_1,e\}} &= \{a_1\}. \end{aligned}$$

A slightly more complicated graphical criterion is a sufficient condition, though. In more detail, if $\mathcal{A}_3$ graphically seperates $\mathcal{A}_1$ and $\mathcal{A}_2$ in $\mathrm{DP}(\mathcal{P})$, and if the program is stratified in a lower layer $\mathcal{A}_3$ and a higher layer $\mathcal{A}_1 \cup \mathcal{A}_2$, then the conditional independency $\mathcal{A}_1 \perp\!\!\!\perp_{\mathcal{P}} \mathcal{A}_2 \mid \mathcal{A}_3$ holds:

Proposition 12. *Let a nlp* $\mathcal{P}$ *with* $\mathrm{DP}(\mathcal{P}) = \langle \mathcal{A}_{\mathcal{P}}, V \rangle$ *be given s.t. the following conditions hold*

1. *there is some* $\mathcal{A}_3 \subseteq \mathcal{A}_{\mathcal{P}}$ *s.t.* $\langle \mathcal{A}_{\mathcal{P}} \setminus \mathcal{A}_3, V \cap ((\mathcal{A}_{\mathcal{P}} \setminus (\mathcal{A}_3) \times (\mathcal{A}_{\mathcal{P}} \setminus (\mathcal{A}_3)) \rangle$ *consists of two disconnected subgraphs* $\langle \mathcal{A}_1, V \cap (\mathcal{A}_1 \times \mathcal{A}_1) \rangle$ *and* $\langle \mathcal{A}_2, V \cap (\mathcal{A}_2 \times \mathcal{A}_2) \rangle$*, and*
2. *for every* $a \in \mathcal{A}_3$ *and* $b \in \mathcal{A}_i$ *(*$i = 1, 2$*),* $b <_{\mathrm{dep}} a$*.*

Then $\mathcal{A}_1 \perp\!\!\!\perp_{\mathcal{P}} \mathcal{A}_2 \mid \mathcal{A}_3$ *holds.*

A case in point of the criteria in this proposition is Example 1. The search for more comprehensive, potentially even necessary, criteria for identifying conditional independencies are an avenue for future work.

7 Related Work

In this section, we provide a summary of related work. For most of the comparisons, we provide more formal comparisons in the extended version of this article (Heyninck 2024a). In the context of classical logic, a notion of conditional independence was proposed by Darwiche (1997). Darwiche assumes a database Δ (i.e. a set of propositional formulas), which is used as a background theory for inferences. The idea behind conditional independence is then that a database Δ sanctions the independence of two sets of atoms x_1 and x_2 conditional on a third set of atoms x_3 if, given full information about x_3, inferences about x_1 are independent from any information about x_2. Our notion of conditional independence implies Darwiche's notion (Heyninck 2024a).

A concept related to conditional independence studied in approximation fixpoint theory is that of *stratification* (Vennekens, Gilis, and Denecker 2006). This work essentially generalizes the idea of *splitting* as known from logic programming, where the idea is to divide a logic program in layers such that computations in a given layer only depend on rules in the layer itself or layers below. For example, the program $\{q \leftarrow \mathrm{not}\ r; r \leftarrow \mathrm{not}\ s; s \leftarrow \mathrm{not}\ p\}$ can be stratified in the layers $\{p\}, \{s, r\}, \{q\}$. This concept was formulated purely algebraically by Vennekens, Gilis, and Denecker (2006). Our study of conditional independence took inspiration from this work in using product lattices as an algebraic tool for dividing lattices, and many proofs and results in our paper are similar to those shown for stratified operators (Vennekens, Gilis, and Denecker 2006). Conceptually, stratification and conditional independence seem somewhat orthogonal, as conditional independence allows to divide a lattice "horizontally" into independent parts, whereas stratification allows to divide a lattice "vertically" in layers that incrementally depend on each other. However, conditional independence can be seen as a special case of stratification (Heyninck 2024a).

A lot of work exists on the *parametrization* of the computational complexity of various computational tasks using *treewidth decompositions* as a parameter (Gottlob, Scarcello, and Sideri 2002). These results show that the computational effort required in solving a problem is not a function of the overall size of the problem, but rather of certain structural parameters of the problem, i.e. the treewidth of a certain representation of the problem. These techniques have been applied to answer set programming (Fichte et al. 2017). In these works, the treewidth of the tree decomposition of the dependency graph $\mathsf{DP}(\mathcal{P})$ and incidence graph (which also contains vertices for rules) of a logic program are used as parameters to obtain fixed-parameter tractability results. Conditional independence and treewidth are orthogonal (Heyninck 2024a). Furthermore, our results go beyond the answer set semantics, e.g. partial stable, supported and well-founded (ultimate) semantics.

Other operator-based formalisms have been analysed in terms of treewidth decompositions (Fichte, Hecher, and Schindler 2022; Dvořák, Pichler, and Woltran 2012). A benefit of our operator-based approach is that all results are purely algebraic and therefore language-independent, which means that applications to specific formalisms are derived as straightforward corollaries. Furthermore, the results for AFT-based semantics, which subsume many KR-formalisms (an overview is provided by (Heyninck and Bogaerts 2023b)), are not restricted to the total stable fixpoints, but also apply to partial stable and well-founded semantics, in contrast to many studies on fixed-parameter tractability.

Conditional independence has been investigated in several other logic-based frameworks, such as (iterated) belief revision (Lynn, Delgrande, and Peppas 2022; Kern-Isberner, Heyninck, and Beierle 2022), conditional logics (Heyninck, Kern-Isberner, and Meyer 2022) and formal argumentation (Rienstra et al. 2020; Gaggl, Rudolph, and Strass 2021). The benefit of our work is that the algebraic nature allows for the straightforward application to other formalisms with a fixpoint semantics.

8 Conclusion

In this paper, the concept of conditional independence, well-known from probability theory, was formulated and studied for operators. This allows to use this concept to a wide variety of formalisms for knowledge representation that admit an operator-based characterisation. As a proof-of-concept, we have applied it to normal logic programs.

There exist several fruitful avenues for future work. Firstly, we want to investigate related notions of independence, such as context-specific independence (Boutilier et al. 1996). A second avenue for future work is a more extensive application of the theory to concrete formalisms, both in breadth (by applying the theory to further formalisms) and in depth (e.g. by investigating more syntactic methods to identify conditional independencies, and by evaluating the computational gain experimentally).

Acknowledgements

I thank Hannes Straß and Johannes P. Wallner for interesting discussions on this topic. I thank the reviewers of KR 2024, to which a previous version of this paper was submitted, for their dilligent reviewing and constructive feedback, as well as the reviewers of this conference. This work was partially supported by the project LogicLM: Combining Logic Programs with Language Model with file number NGF.1609.241.010 of the research programme NGF AiNed XS Europa 2024-1 which is (partly) financed by the Dutch Research Council (NWO).

References

Bogaerts, B. 2015. *Groundedness in logics with a fixpoint semantics*. Ph.D. thesis, Informatics Section, Department of Computer Science, Faculty of Engineering Science. Denecker, Marc (supervisor), Vennekens, Joost and Van den Bussche, Jan (cosupervisors).

Bogaerts, B.; and Jakubowski, M. 2021. Fixpoint Semantics for Recursive SHACL. In Formisano, A.; Liu, Y. A.; Bogaerts, B.; Brik, A.; Dahl, V.; Dodaro, C.; Fodor, P.; Pozzato, G. L.; Vennekens, J.; and Zhou, N., eds., *Proceedings 37th International Conference on Logic Programming (Technical Communications), ICLP Technical Communications 2021, Porto (virtual event), 20-27th September 2021*, volume 345 of *EPTCS*, 41–47.

Boutilier, C.; Friedman, N.; Goldszmidt, M.; and Koller, D. 1996. Context-specific independence in Bayesian networks. In *Proc. 12th Conf. on Uncertainty in Artificial Intelligence (UAI'96)*, 115–123.

Cousot, P.; and Cousot, R. 1979. Constructive versions of Tarski's fixed point theorems. *Pacific journal of Mathematics*, 82(1): 43–57.

Darwiche, A. 1997. A logical notion of conditional independence: properties and applications. *Artificial Intelligence*, 97(1-2): 45–82.

Davey, B. A.; and Priestley, H. A. 2002. *Introduction to lattices and order*. Cambridge university press.

Denecker, M.; Bruynooghe, M.; and Vennekens, J. 2012. Approximation fixpoint theory and the semantics of logic and answers set programs. In *Correct reasoning*, 178–194. Springer.

Denecker, M.; Marek, V.; and Truszczyński, M. 2000. Approximations, stable operators, well-founded fixpoints and applications in nonmonotonic reasoning. In *Logic-based Artificial Intelligence*, volume 597 of *The Springer International Series in Engineering and Computer Science*, 127–144. Springer.

Denecker, M.; Marek, V.; and Truszczyński, M. 2003. Uniform semantic treatment of default and autoepistemic logics. *Artificial Intelligence*, 143(1): 79–122.

Denecker, M.; Marek, V. W.; and Truszczynski, M. 2002. Ultimate approximations in nonmonotonic knowledge representation systems. In *Proceedings of the Eights International Conference on Principles of Knowledge Representation and Reasoning*, 177–190.

Denecker, M.; Marek, V. W.; and Truszczyński, M. 2004. Ultimate approximation and its application in nonmonotonic knowledge representation systems. *Information and Computation*, 192(1): 84–121.

Downey, R. G.; and Fellows, M. R. 2013. *Fundamentals of parameterized complexity*, volume 4. Springer.

Dvořák, W.; Pichler, R.; and Woltran, S. 2012. Towards fixed-parameter tractable algorithms for abstract argumentation. *Artificial Intelligence*, 186: 1–37.

Fichte, J. K.; Hecher, M.; Morak, M.; and Woltran, S. 2017. Answer set solving with bounded treewidth revisited. In *Logic Programming and Nonmonotonic Reasoning: 14th International Conference, LPNMR 2017, Espoo, Finland, July 3-6, 2017, Proceedings 14*, 132–145. Springer.

Fichte, J. K.; Hecher, M.; and Schindler, I. 2022. Default logic and bounded treewidth. *Information and Computation*, 283: 104675.

Fitting, M. 1991. Bilattices and the semantics of logic programming. *The Journal of Logic Programming*, 11(2): 91–116.

Gaggl, S. A.; Rudolph, S.; and Strass, H. 2021. On the decomposition of abstract dialectical frameworks and the complexity of naive-based semantics. *Journal of Artificial Intelligence Research*, 70: 1–64.

Gottlob, G.; Scarcello, F.; and Sideri, M. 2002. Fixed-parameter complexity in AI and nonmonotonic reasoning. *Artificial Intelligence*, 138(1-2): 55–86.

Heyninck, J. 2024a. An Algebraic Notion of Conditional Independence, and Its Application to Knowledge Representation (full version). arXiv:2412.13712.

Heyninck, J. 2024b. Operator-based semantics for choice programs: is choosing losing? (full version).

Heyninck, J.; Arieli, O.; and Bogaerts, B. 2022. Non-Deterministic Approximation Fixpoint Theory and Its Application in Disjunctive Logic Programming. *CoRR*, abs/2211.17262.

Heyninck, J.; and Bogaerts, B. 2023a. Non-deterministic Approximation Operators: Ultimate Operators, Semi-equilibrium Semantics, and Aggregates. *Theory Pract. Log. Program.*, 23(4): 632–647.

Heyninck, J.; and Bogaerts, B. 2023b. Non-deterministic approximation operators: ultimate operators, semi-equilibrium semantics and aggregates (full version). *CoRR*, abs/2305.10846.

Heyninck, J.; Kern-Isberner, G.; and Meyer, T. 2022. Conditional Syntax Splitting, Lexicographic Entailment and the Drowning Effect.

Heyninck, J.; Kern-Isberner, G.; Meyer, T. A.; Haldimann, J.; and Beierle, C. 2023. Conditional syntax splitting for non-monotonic inference operators. In *Proceedings of the 37th AAAI Conference on Artificial Intelligence (AAAI'23)*.

Kern-Isberner, G.; Heyninck, J.; and Beierle, C. 2022. Conditional independence for iterated belief revision. In *31st International Joint Conference on Artificial Intelligence*, 2690–2696. International Joint Conferences on Artificial Intelligence.

Lang, J.; Liberatore, P.; and Marquis, P. 2002. Conditional independence in propositional logic. *Artificial Intelligence*, 141(1-2): 79–121.

Liu, F.; and You, J.-H. 2022. Alternating fixpoint operator for hybrid MKNF knowledge bases as an approximator of AFT. *Theory and Practice of Logic Programming*, 22(2): 305–334.

Lynn, M. J.; Delgrande, J. P.; and Peppas, P. 2022. Using conditional independence for belief revision. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 5809–5816.

Makinson, D.; and van der Torre, L. 2003. What is input/output logic? In *Foundations of the Formal Sciences II: Applications of Mathematical Logic in Philosophy and Linguistics, Papers of a Conference held in Bonn, November 10–13, 2000*, 163–174. Springer.

Pearl, J.; Geiger, D.; and Verma, T. 1989. Conditional independence and its representations. *Kybernetika*, 25(7): 33–44.

Pelov, N.; Denecker, M.; and Bruynooghe, M. 2007. Well-founded and stable semantics of logic programs with aggregates. *Theory and Practice of Logic Programming*, 7(3): 301–353.

Przymusinski, T. C. 1990. The well-founded semantics coincides with the three-valued stable semantics. *Fundamenta Informaticae*, 13(4): 445–463.

Rienstra, T.; Thimm, M.; Kersting, K.; and Shao, X. 2020. Independence and D-separation in Abstract Argumentation. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, volume 17, 713–722.

Strass, H. 2013. Approximating operators and semantics for abstract dialectical frameworks. *Artificial Intelligence*, 205: 39–70.

Strass, H.; and Wallner, J. P. 2015. Analyzing the Computational Complexity of Abstract Dialectical Frameworks via Approximation Fixpoint Theory. *Artificial Intelligence*, 226: 34–74.

Van Gelder, A.; Ross, K. A.; and Schlipf, J. S. 1991. The well-founded semantics for general logic programs. *Journal of the ACM*, 38(3): 619–649.

Vennekens, J.; Gilis, D.; and Denecker, M. 2006. Splitting an operator: Algebraic modularity results for logics with fixpoint semantics. *ACM Transactions on computational logic (TOCL)*, 7(4): 765–797.